FITS files (structure):

A FITS file contains several so called *Extensions*. One extension is compiled of

One header:

- Stored as ascii data
- User defined values (8 character descriptor, ~70 characters for data (int, float, string, etc) plus an optional comment)
- Forward and backward compatible (new values, missing values)
- Order doesn't matter
- Table definition (e.g. Byte/Int/Float/Float/Float/char[40]/Float[10])

One data block (“table”):

- Stored as binary data, row by row
- N rows with data according to the definition above

The DAL (data access layer) library allows to setup files with a similar structure than root-file, i.e. so called Groups can point to other Groups which contain the extensions.

In this context the *pointers* can point to other tables, other files or even a web URL (implementing a database interface would also be possible)

This allows to have several files containing the data, but only one file as an input to the program.

FITS (pro and con):

Disadvantages:

- Mixing of several kind of data in one file is difficult and can be slow
- Consequently, the file structure (what is a header, what is in a table, what are individual files) to be chosen is more a consequence of performance issues than a very good structure
  → most probably one file per subsystem-report (e.g. Drive, Starguider, etc)

Advantages:

- Well defined format, used by a large community
- Not worse than current MAGIC format, but a standard
- More flexible than current MAGIC format
- A *politically correct* solution
- I/O libarary (cfitsio, dal) available, proven to work well and well maintained
- The file structure could be similar to the MAGIC data → should not be slower
- A lot of experience available at ISDC (ISDC could help a lot with the definition and maybe also with the implementation)
- A variety of plotting tools available, e.g. to plot Zenith Angle vs. Time from the drive-reports
  → everybody can have a quick look into the file contents and do simple plots
- Implementation into MARS should be fast and straight forward
- Easy to maintain later, simple to enhance

EXAMPLE

Access handles transparently
by libDAL and libFITSIO